

Introduction to Database Systems

CSE 444

Lecture 23: Pig Latin (continued)

Where we are...

- Previously...
 - LOAD – read data
 - FOREACH – with and without flatten
 - FILTER
 - JOIN
 - GROUP BY
- Now...
 - COGROUP: A generic way to group tuples from two datasets together

Co-Group

Dataset 1 results: {(queryString, url, position)}

Dataset 2 revenue: {(queryString, adSlot, amount)}

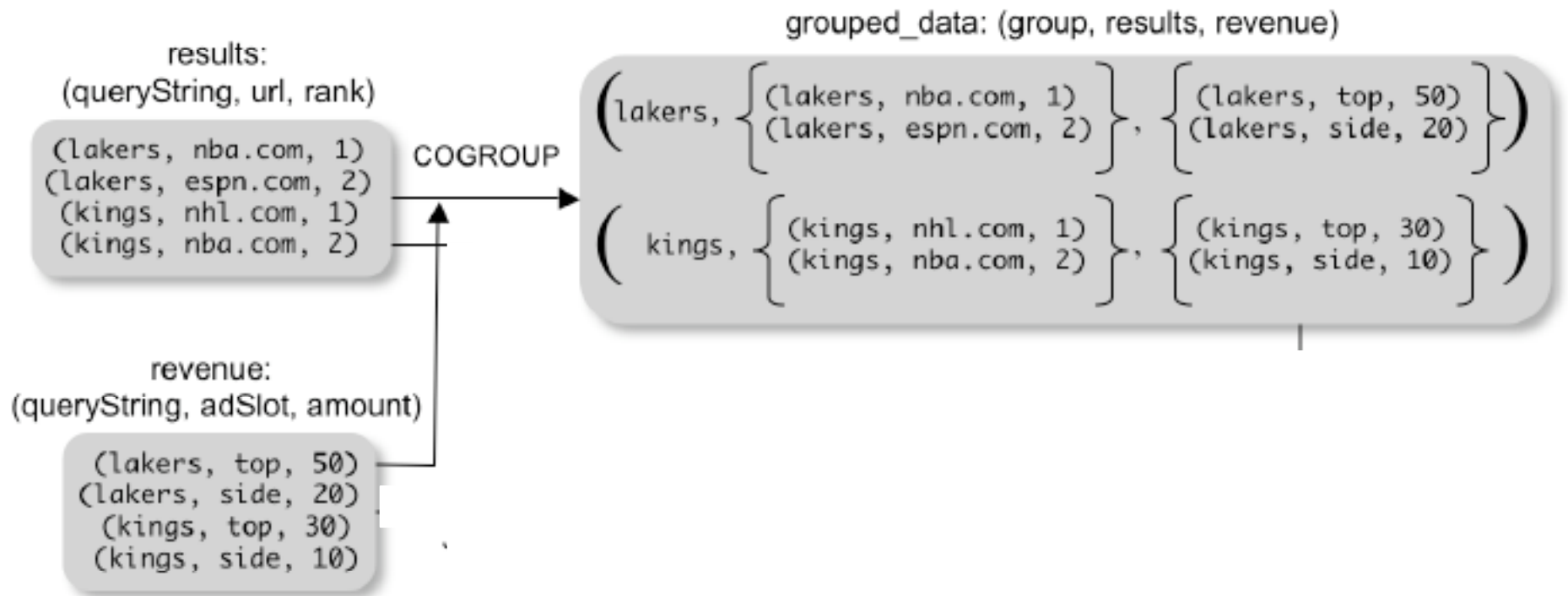
```
grouped_data =  
    COGROUP results BY queryString,  
    revenue BY queryString;
```

```
grouped_data: {(queryString, results: {(url, position)},  
               revenue: {(adSlot, amount)}}}
```

What is the output type in general ?

```
{group_id, bag dataset 1, bag dataset 2}
```

Co-Group



Is this an inner join or an outer join ?

Co-Group

```
grouped_data: {(queryString, results:{(url, position)},  
               revenue:{(adSlot, amount)}}}
```

```
url_revenues = FOREACH grouped_data  
  GENERATE  
    FLATTEN(distributeRevenue(results, revenue));
```

...where `distributeRevenue` is a UDF that accepts search results and revenue information for a query string at a time, and outputs a bag of urls and the revenue attributed to them.

Co-Group v.s. Join

```
grouped_data: {(queryString, results:{(url, position)},  
               revenue:{(adSlot, amount)})}
```

```
grouped_data = COGROUP results BY queryString,  
               revenue BY queryString;  
join_result = FOREACH grouped_data  
               GENERATE FLATTEN(results),  
               FLATTEN(revenue);
```

Result is the same as JOIN

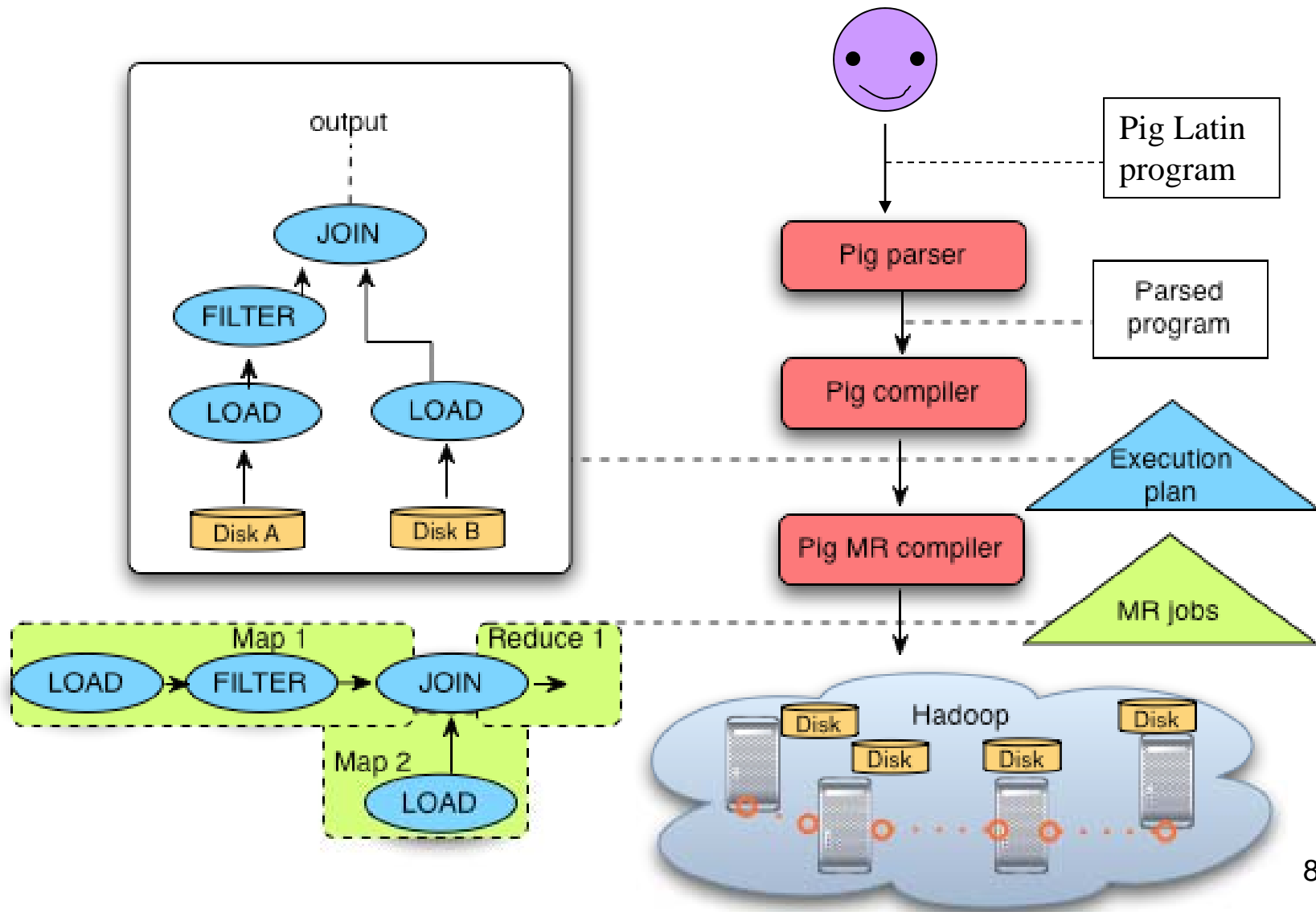
Asking for Output: STORE

```
STORE query_revenues INTO `theoutput`  
    USING myStore();
```

Meaning: write query_revenues to the file 'theoutput'

This is when the entire query is finally executed!

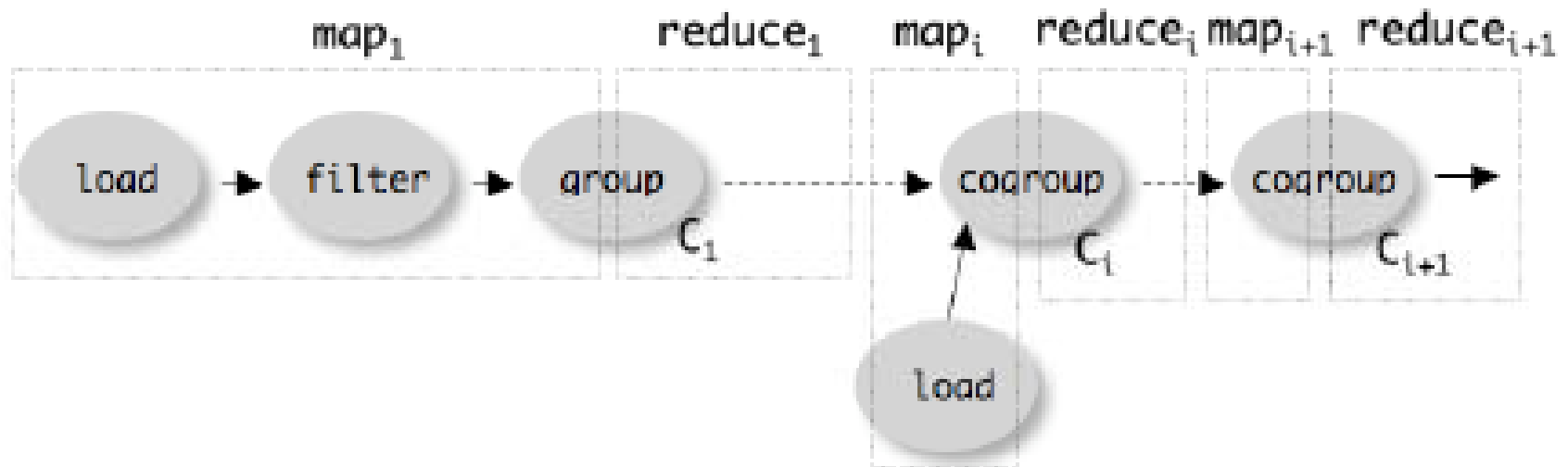
Query Processing Steps



Implementation

- Over Hadoop
- Parse query:
 - All between LOAD and STORE → one logical plan
- Logical plan → ensemble of MapReduce jobs
 - Each (CO)Group becomes a MapReduce job
 - Other ops merged into Map or Reduce operators
- Extra MapReduce jobs for sampling before SORT operations

Implementation



Advice for the Project

- Always run first locally
 - Test your program on your local machine, on a smaller dataset
 - After you debugged the program, send it to the cluster
- Batch processing:
 - Keep in mind that Hadoop does batch processing
 - Your job takes 2-7 minutes on the cluster
 - No-one else can run on the same compute nodes during this time !!

Longer Example: Tutorial Script 2

- Goal: Process a search query log file and find search phrases that occur with particular high frequency during certain times of the day

```
raw = LOAD 'excite-small.log'  
      USING PigStorage('\t') AS (user, time, query);
```

```
clean1 = FILTER raw BY  
         org.apache.pig.tutorial.NonURLDetector(query);
```

Longer Example: Tutorial Script 2

```
clean2 = FOREACH clean1
    GENERATE user, time,
    org.apache.pig.tutorial.ToLower(query)
    as query;

houred = FOREACH clean2
    GENERATE user,
    org.apache.pig.tutorial.ExtractHour(time)
    AS hour, query;
```

Longer Example: Tutorial Script 2

```
ngramed1 = FOREACH houred  
            GENERATE user, hour,  
            flatten(org.apache.pig.tutorial.NGramGenerator(query))  
            AS ngram;
```

```
ngramed2 = DISTINCT ngramed1;
```

```
hour_frequency1 = GROUP ngramed2 BY (ngram, hour);  
hour_frequency2 = FOREACH hour_frequency1  
            GENERATE flatten($0), COUNT($1) as count;
```

Longer Example: Tutorial Script 2

```
hour_frequency3 = FOREACH hour_frequency2  
  GENERATE $0 as ngram, $1 as hour, $2 as count;
```

```
hour00 = FILTER hour_frequency2 BY hour eq '00';  
hour12 = FILTER hour_frequency3 BY hour eq '12';
```

```
same – JOIN hour00 BY $0, hour12 BY $0;
```

Longer Example: Tutorial Script 2

```
same1 = FOREACH same
```

```
  GENERATE hour_frequency2::hour00::group::ngram  
  AS ngram, $2 as count00, $5 as count12;
```

```
STORE same1
```

```
  INTO 'script2-local-results.txt' USING PigStorage();
```